**Final Assignment**

**Max Points: 100**

You are tasked with creating a relational database for online stock trading. Here are two links to help you get started:

1. [Very Basic Introduction to Stock Trading](http://invest-faq.com/articles/trade-a-basics.html)
2. [E\*TRADE Conditional Orders Tutorial](https://us.etrade.com/e/t/invest/apptemplate?gxml=conditional_tutorial.html&skinname=none&rightrail=disable)

The basic idea behind your on-line trading database is that it will allow customers to browse/search the contents of your database (at least that part you want the customer to see) and to trade stocks. Your database must be based on the specifications and requirements that follow.

# System Users

The users of your system will be: (i) Customers that use your system to trade stocks and pay fees for doing so; (ii) Customer representatives who provide customer-related services; and (iii) Database manager. Customers of a stock-trading system are sometimes also known as *clients*, so I shall use these two terms interchangeably.

**Required Data**

The data items required for the stock-trading database can be classified into four categories: *orders*, *stocks*, *customers* and *employees*, where an *order* is an order to buy or sell a certain number of shares of a particular stock at a certain price. This classification does not imply any particular table arrangement. You are responsible for arranging the data items into tables, determining the relationships among tables and identifying the key attributes. Finally, you should specify and enforce integrity constraints on the data, including referential integrity constraints. As I mentioned in class, you will first create an E-R diagram of your online trading database before developing your relational model.

**Order Data**

This category of data should include the following items:

* 1. Order ID
  2. Stock Symbol(GM, GE, IBM, etc.)
  3. Order Type (Buy, Sell)
  4. Number of Shares
  5. Customer Account Number (of the buyer or seller)
  6. Date/Time (the order was placed)
  7. Transaction Fee
  8. Price Type (Market, Market on Close, Trailing Stop, Hidden Stop)
  9. Employee ID

An order is the mechanism a customer uses to buy or sell a certain number of shares of a particular stock at a certain price. A transaction fee of 5% is associated with every order.

Your online trading database will also support [Conditional Orders](https://us.etrade.com/e/t/invest/apptemplate?gxml=conditional_tutorial.html&skinname=none&rightrail=disable) such as a *Trailing Stop* or *Hidden Stop* order. A Trailing-Stop order uses a trailing stop to automatically decide when to place a sell order. A trailing stop is set at a percentage or dollar amount below the stock's current market price, and a sell order will be placed if and when the share price falls to the trailing stop. The trailing stop is adjusted as the share price fluctuates.

In contrast, a Hidden-Stop order uses a fixed hidden-stop price to automatically decide when to place a sell order. That is, a sell order will be placed if and when the stock price falls to the hidden stop.

To understand conditional orders better, suppose that you have just bought 1000 shares of GM at $50.00, and you decide that you only want to risk $5.00 per share on this transaction. Accordingly, you immediately place a hidden-stop order at $45.00. This means that if the price of GM should drop to $45.00, your broker will sell your 1000 shares at a market price of $45.00. The use of a hidden-stop order will therefore pre-determine the maximum loss a trader will incur.

Instead of placing a hidden-stop order on your GM shares, suppose now that you place a trailing-stop order with a trailing-stop value of $5.00 (or, equivalently, a trailing-stop percentage of 10%). Thus, your shares will be sold if the share price drops to $45.00. But instead of declining, the price of GM increases to $60.00; but so does the trailing stop. So now your shares will be sold if the share price drops to $55.00. Thus the trailing-stop technique allows an investor to set a limit on the maximum possible loss without setting a limit on the maximum possible gain, and without requiring paying attention to the investment on an ongoing basis.

To find out more about how conditional orders work, please consult the [E\*TRADE Conditional Orders Tutorial](https://us.etrade.com/e/t/invest/apptemplate?gxml=conditional_tutorial.html&skinname=none&rightrail=disable).

**Stock Data**

This category of data should include the following items:

* 1. Stock Symbol
  2. Stock Name
  3. Stock Type
  4. Share Price
  5. Number of Shares

An order involves the purchase or sale of a certain number of shares of a stock at a certain price. Stocks are of a certain type: GM is an automotive stock, IBM is a computer stock, etc. You can populate your database with any kind of stocks you like.

**Customer Data**

The items required for this category include:

* 1. Last Name
  2. First Name
  3. Address
  4. City
  5. State
  6. Zip Code
  7. Telephone
  8. E-mail Address
  9. Account Number
  10. Account Creation Date
  11. Credit Card Number
  12. Stock Portfolio
  13. Rating

A given customer may partake in any number of stock transaction, either as a buyer or as a seller. A customer may have one or more accounts from which to trade stocks. Associated with each account is a stock portfolio, indicating which stocks (and number of shares) are held in that account. The customer's rating should reflect how active a trader he or she is.

**Employee Data**

This category of data should include the following:

* 1. Social Security #
  2. Last Name
  3. First Name
  4. Address
  5. City
  6. State
  7. Zip Code
  8. Telephone
  9. Start Date
  10. Hourly Rate
  11. Access Level (Manager, Customer Representative)

**Transactions**

  A database *transaction* can be viewed as a small program (written in DML) that either updates or queries the database. Transactions that change the contents of the database must do so in a consistent manner. What follows is a breakdown of the user-level transactions that your database system should support.

**Manager-Level Transactions**

The manager should be able to:

* Set the share price of a stock (for simulating market fluctuations in a stock's share price)
* Add, Edit and Delete information for an employee
* Obtain a sales report for a particular month
* Produce a comprehensive listing of all stocks
* Produce a list of orders by stock symbol or by customer name
* Produce a summary listing of revenue generated by a particular stock, stock type, or customer
* Determine which customer representative generated most total revenue
* Determine which customer generated most total revenue
* Produce a list of most actively traded stocks

**Customer-Representative-Level Transactions**

Customer Representatives should be thought of as stock brokers and should be able to:

* Record an order
* Add, Edit and Delete information for a customer
* Produce customer mailing lists
* Produce a list of stock suggestions for a given customer (based on that customer's past orders)

**Customer-Level Transactions**

Customers should be thought of as online traders and should be able to place orders to purchase or sell stocks. In particular, they should be able to place a trailing-stop or hidden-stop conditional order, and place an order to buy or sell stocks at market or close-of-market price. While they will not be permitted to access the database directly, they should be able to retrieve the following information:

* A customer's current stock holdings
* The share-price and trailing-stop history for a given conditional order
* The share-price and hidden-stop history for a given conditional order
* The share-price history of a given stock over a certain period of time (e.g., past six months)
* A history of all current and past orders a customer has placed
* Stocks available of a particular type and most-recent order info
* Stocks available with a particular keyword or set of keywords in the stock name, and most-recent order info
* Best-Seller list of stocks
* Personalized stock suggestion list

# Your Tasks

1. **Entity-Relationship (E-R) Diagram of the complete database scheme.**

Diagram

Description automatically generated

**2. Lucid description of the relational database scheme for your online stock trading database, based on your ER diagram. Include a discussion of the reasoning behind your design decisions (e.g., how did you normalize?).**

**Solution –**

Person, a superclass of both Customers and Employees, is a newly generated entity. Both classes' shared characteristics are present in Person, preventing data duplication. Both the employee and the customer have distinguishing characteristics. Since each consumer and employee has a different identification number, CusId and EmpId are the main keys to their respective entities. A customer may have any number of accounts, but each account must be associated with just one customer, and each customer must have at least one account. As a result, the relationship between Customer and HasAccount has a bold line denoting a participation constraint to the Account, but the relationship between Account and HasAccount has a bolded arrow denoting a participation and key constraint to the Customer.

The primary key, which is different for each account, serves as its representation. Each account has a stock portfolio, which refers to the collection of equities it has. There are no restrictions because an account with no stocks does not have a stock portfolio. A bolded arrow pointing to HasPortfolio denotes the participation and key constraint from Portfolio to Account that results from a portfolio being linked to only one account. There are no restrictions from portfolio to entity stock; portfolio may comprise any number of stocks or none at all. There are no restrictions from stock to portfolio; a stock may be a part of one portfolio, many portfolios, or none at all. Every share of stock has a different symbol, which is Stock's primary key. Customers and staff can each place a stock order, but neither is required to do so. To represent the customers and staff placing stock orders, the relationship Order is created. OrderId serves as the primary key because each Order has a distinct Id number. The bolded participation constraint arrow indicates that the customer and employees each have a unique set of credentials. The username, password, ID, and Account type make up these credentials. Because both employees and customers are required to have distinct usernames, the username serves as the main key for login. Entities Transact, and any number of Orders can make up a ConditionalPriceHistory. OrderId is the main key of Order, hence they are foreign keys with referential integrity constraints.

Since each transaction is distinct, Transact has Id as its main key. Conditional Price History has  OrderId, PriceType, and Timestamp as its primary keys. This is because unique entries are denoted by the order id, its price type, whether trailing stop or hidden stop, and the unique time at which the Order was recorded into the table. An entity comprised of stocks is called StockHistory. A StockHistory entity has the attributes timestamp and Share price, which monitor when the stock price changed and what the new price was, and it references a specific StockSymbol, making it a component of the primary key. Any number of Stocks may be used to create a StockHistory.

**3. A list of all functional dependencies in the relational database scheme.**

**Solution -**

1. **ConditionalPriceHistory**

OrderId, PriceType, TimeStamp → CurSharePrice, StopPrice

1. **Portfolio**

AccNum, StockSymbol → NumShares, Stop, StopPrice

1. **StockPriceHistory**

StockSymbol, TimeStamp → SharePrice

1. **Transact**

Id → OrderId, Transfee, TimeStamp

1. **Customer**

CusId → LastName, FirstName, Address, City, State, ZipCode, Telephone, Email, Rating, Email

1. **Employee**

EmpId → SSN, LastName, FirstName, Address, City, State, ZipCode, Telephone, Start Date, HourlyRate, Position

SSN → LastName, FirstName, Address, City, State, ZipCode, Telephone, StartDate, HourlyRate, Position, EmpId

1. **Stock**

StockSymbol → StockName, StockType, SharePrice, NumAvailShares

StockName → StockType, SharePrice, NumAvailShares

1. **Account**

AccNum → AccCreDate, CreditNum, CusId

1. **Order**

OrderId → StockSymbol, OrderType, NumShares, CusAccNum, TimeStamp, PriceType, StopPrice, StopDiff, CurSharePrice, EmpId, Recorded, Completed

StockSymbol, TimeStamp, CusAccNum, EmpId → OrderType, NumShares, PriceType, StopPrice, StopDiff, CurSharePrice, Recorded, Completed

**4.Description of integrity constraints including referential integrity.**

Integrity constraints are rules that must be followed. It is used to keep information of high quality. Integrity restrictions ensure that data insertion, updating, and other activities are carried out in a way that does not compromise data integrity. Thus, integrity constraint is employed to prevent against unintended harm to the database.

**Types of Integrity Constraints:**

* **Domain constraints -** Domain constraints are the specification of a valid set of values for an attribute. Domain data types include string, character, integer, time, date, currency, and so on. The attribute's value must be available in the relevant domain.

Here, the domain constraints are maintained because each attribute in the table has it’s specific domain datatypes. For instance, in table STOCK, the SharePrice is a float value, name, type and symbol of the stocks is varchar. Similarly, the domain constraints are followed by all the other tables that are created.

* **Entity integrity constraints -** According to the entity integrity constraint, the primary key value cannot be null. This is due to the fact that the primary key value is used to identify particular rows in a relation, and if the primary key is null, we cannot identify those rows. A table can include a null value other than the primary key field.

Here, the entity integrity constraint is maintained because each value in the primary key in all the tables is not a NULL value.

* **Referential Integrity Constraints** - Between two tables, a referential integrity constraint is given. If a foreign key in Table 1 relates to the Primary Key in Table 2, then every value of the Foreign Key in Table 1 must be null or available in Table 2.

Here, the referential integrity constraint is maintained because each value in the foreign key of the table relates to the primary key of the corresponding table. Also, the Foreign Key is not a NULL value and is available.

* **Key constraints** - Keys are the entity set used to uniquely identify an entity within its entity set. An entity set can have numerous keys, but only one of them is the primary key. In a relational table, a primary key might have both a unique and null value.

Here, the key constraints is maintained because an entity in any of the tables can have numerous keys, but only one of them can be a primary key. Example – In the STOCK table, stockSymbol is the primary key.